ITI8700: Knowledge Representation

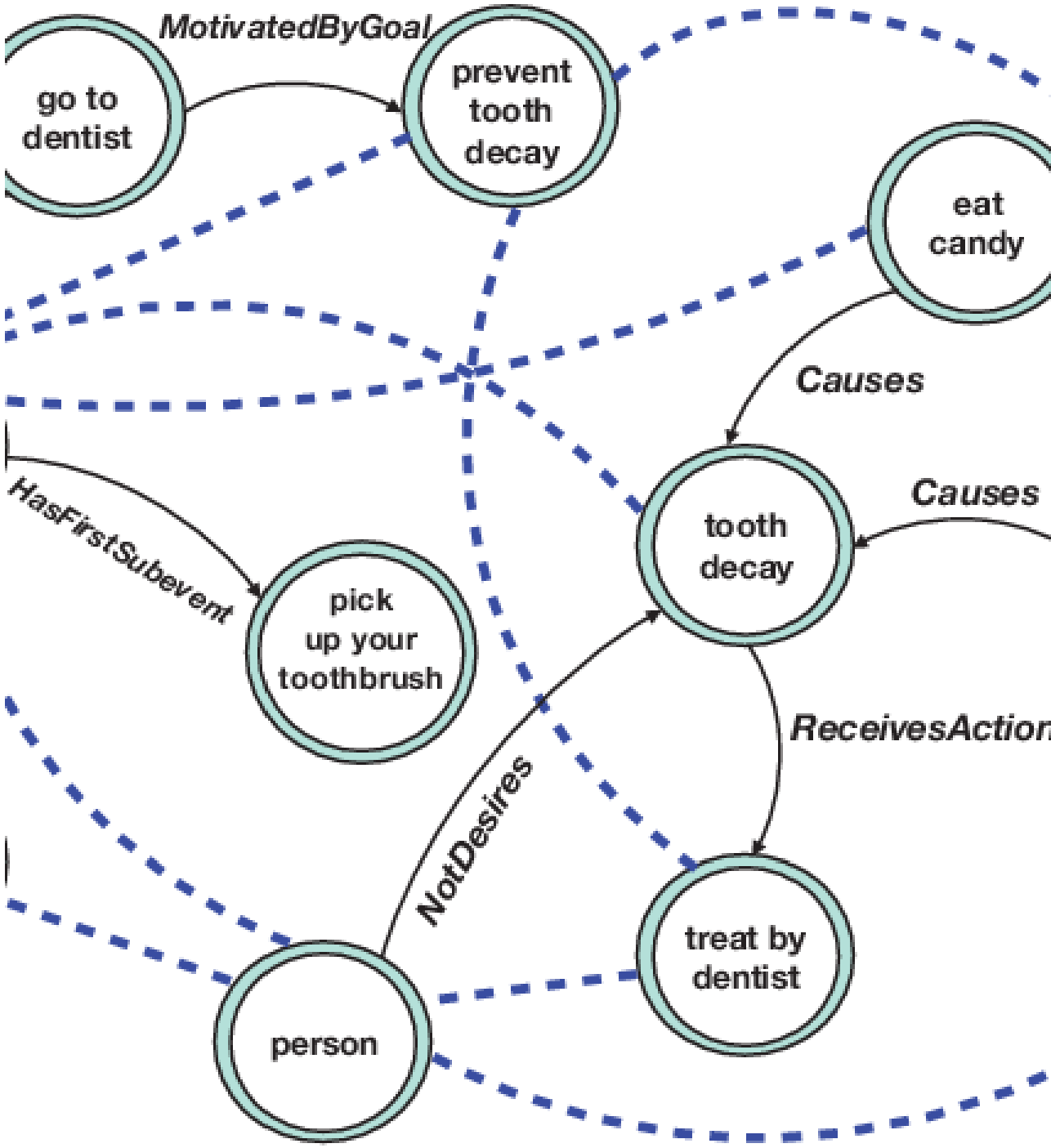
Semantic Parsing: A Practical Approach

Martin Verrev

Spring 2024

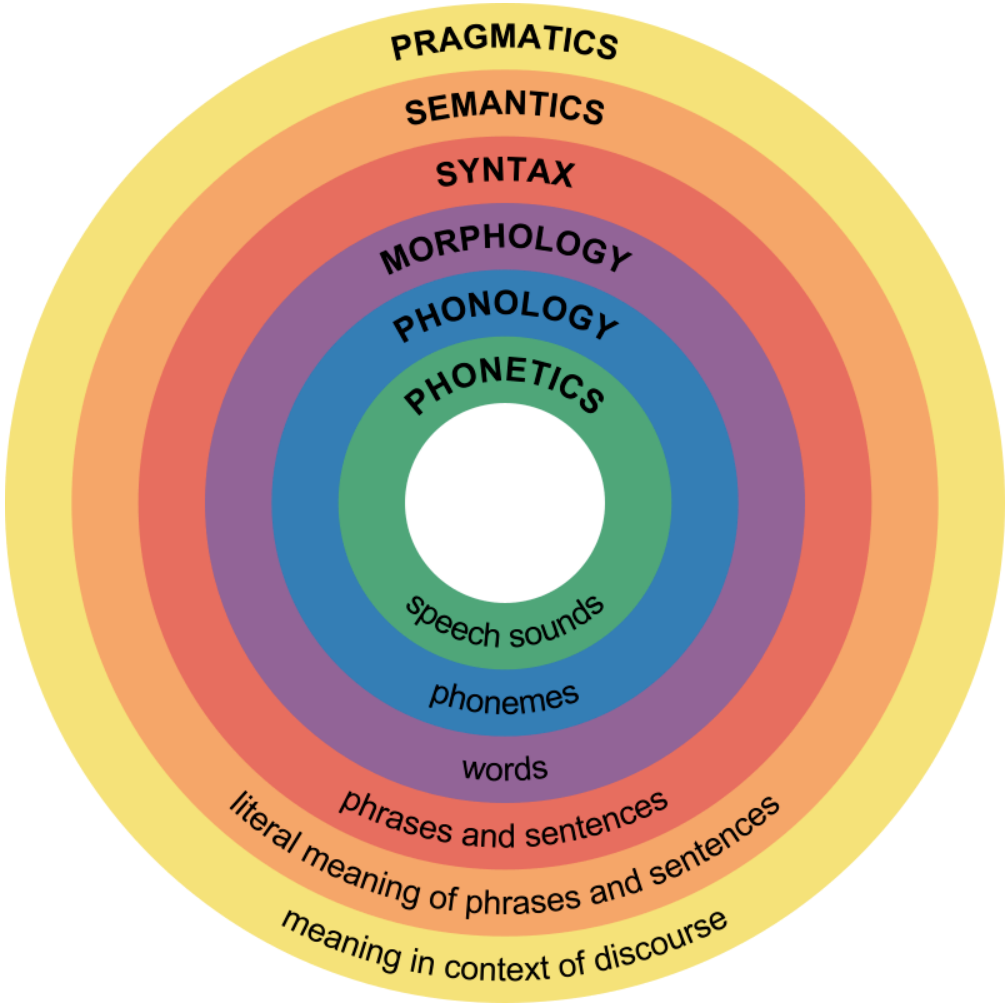
Commonsense Knowledge

Facts about the everyday world that a typical seven-year-old is expected to know including but not limited to space, time, objects, substances, environment, human psychology, societal norms, etc.



Semantic Parsing

The task of converting natural language utterances to formal, machine-understandable representations of their meaning.



AMR

Abstract Meaning Representation is a semantic formalism based on propositional logic and the neo-Davidsonian event representations where each representation is a single-rooted, directed graph. AMR is strongly biased towards English though it does support multilingual meanings. Its concepts are either English verbs, PropBank framesets, or specific keywords. AMR also supports NER, question detection, within-sentence co-reference, modality and question identification.

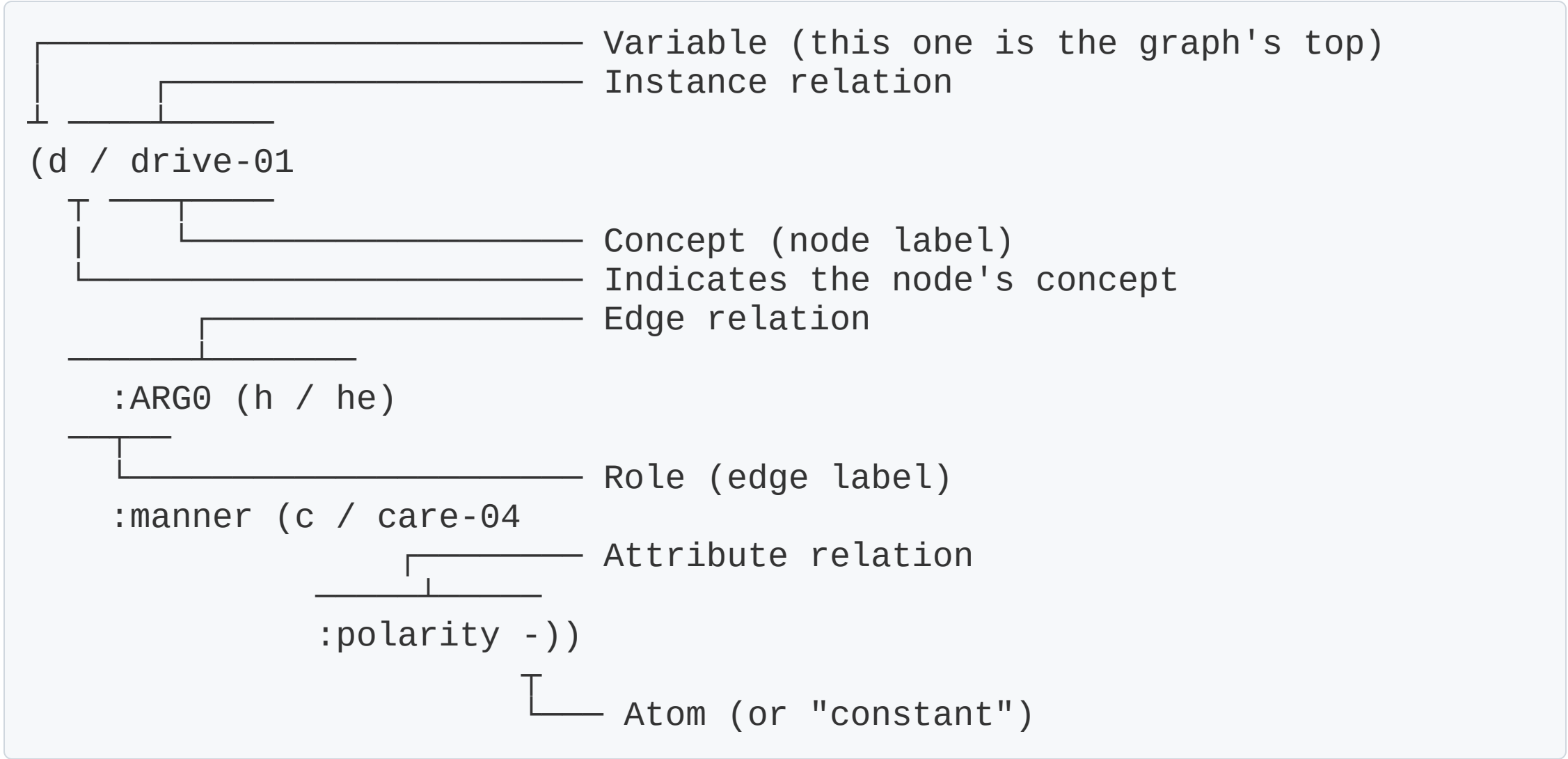
AMR Representations

Benefits of AMR

- Tools for processing: https://penman.readthedocs.io/en/latest/api/penman.html
- Can be represented as triples: manageable post-processing
- Penman notation both human and machine-readable.
- Question detection with `amr-unknown` keyword.
- Additional information extracted as compared to UD:
  - Role reification
  - NER and nunber detection
  - Tone detection
  - Wikification

  .. etc

Penman notation



Example

Brutus stabs Caesar with a knife.

AMR parse graph

```
(s / stab-01
      :ARG0 (p / person
            :name (n / name
                  :op1 "Brutus"))
      :ARG1 (p2 / person
            :name (n2 / name
                  :op1 "Caesar"))
      :instrument (k / knife))
```

Example

Brutus stabs Caesar with a knife.

UD parse

```
root: stab (stabs) upos:VERB xpos:VBZ lemma:stab ↵
        feats:Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin]
  nsubj: Brutus upos:PROPN xpos:NNP lemma:Brutus ner:S-PERSON]
  obj: Caesar upos:PROPN xpos:NNP lemma:Caesar ner:S-PERSON]
  obl: knife upos:NOUN xpos:NN lemma:knife]
    case: with upos:ADP xpos:IN lemma:with]
    det: a upos:DET xpos:DT lemma:a]
```

Example

Brutus stabs Caesar with a knife.

Resulting logical form from *nlpsolver*

```
isa(knife,cs4)
rel2(with,cs3,cs4,ctxt(Pres,1))
isa(agora,cs5)
rel2(in,cs3,cs5,ctxt(Pres,1))
do2(stab,c2_Brutus,c1_Caesar,cs3,ctxt(Pres,1))
```

Can we somehow get additional information to this list of clauses?

Constructing a logical Form

From Jurafsky: 16.4 "Logical Representations of Sentence Meaning: Event and State Representation"

1. Events are captured with predicates that take single event variable as an argument. Events are denoted by verbs.
2. There is no need to specifiy a fixed number of arguments for a given FOL predicate: as many roles and fillers can be glued on, e.g. `stabs(Brutus, Caesar) & with(knife) & in(agora)`
3. No more roles are postulated than are mentioned in input.
4. The logical connections between closely related inputs that share same predicate are satisfied without the need for additional inference.
5. Syntactic arguments form the arguments of semantic predicates.

Constructing a logical form

Construction of logical form consists of the following steps:

1. Find predicates, specify their arguments (number, type)
   1.1 Concepts and events - `predicate/1`
   1.2 Roles - `predicate/2`
2. Construct corresponding atoms
3. Divide atoms on same level into groups
4. Specify connectives between atoms of each group and construct corresponding formulas.

Constructing a logical form

5. Divide formulas and/or any of the remaining atoms of the same level into groups. If there are no groups go to step 7
6. Specify connectives between elements of each group
7. Specify quantifers for the variables.
8. Construct final FOL formula.

Example

Brutus stabs Caesar with a knife.

Clausified Form

```
[
  ['Agent', 'stab-01', 'person'],
  ['Patient', 'stab-01', 'person0'],
  ['instrument', 'stab-01', 'knife'],
  ['hasName', 'person', 'Brutus'],
  ['hasName', 'person0', 'Caesar']
]
```

Simple annotation scheme for encoding AMR information

Titanic sank in the Atlantic in 1912.

```
Titanic {"role": "Patient", "wiki": "RMS_Titanic",
"type": "ship", "cat": "product"} sank {"instance":
"sink-01", "location": "Atlantic", "time": "1912"}
in the Atlantic {"wiki": "Atlantic_Ocean",
"type": "ocean", "cat": "location"} in 1912
{"instance": "date-entity", "year": "1912"}
```

Key	Semantics
instance	Reference to Propbank frame or date entity
role	Propbank core role for frame \newline argument
wiki	Reference to linked entry on English Wikipedia, \newline e.g. { t ``RMS_Titanic''}
type	Entity type, e.g. {\tt ``ship''}
cat	Entity parent category \newline e.g. {\tt ``product''}
ref	Coreference referent, \newline e.g {\tt ``John''} and {\tt ``he''}
polarity	Negative (``-'') or positive (``+'') \newline polarity

Additional information gained:

- Escaping invalid or complex syntax;
- Events and semantic roles;
- Adding context to numeric values;
- Reifying AMR specific abstract roles;
- Fine-grained named entity recognition (NER)

Still Problems with Ambiguity

"The sailor killed the pirate with a sword."

We cannot infer whole meaning without knowing the context.

Constructing the Context

1. Keeping track of sentence order.
2. Keeping track of named entities.
3. Classifying the sentence type:
    - *Conceptual*: Typically, they describe concepts or relations between them. Example: Elephant is a big animal.
    - *Facts*: Named entities not dependent on uncommon circum-stances. Example: Rome is the capital of Italy.
    - *Situational*: A concrete situation and events happening within this situa tion. Example: Brutus stabbed Caesar with a knife.

Work To Do

- **Compound sentence segmentation**. Given compound sentences they are not split before parsing, resulting in increased complexity of generated logical forms (due to connectives).
- __Context order and merging
- **Question detection and scope.** Question parsing is not implemented in the current version of the system.
- **Combine AMR and UD representations in final logical form**
- **.. and many more**

Links

- **amrlib:** A toolkit for AMR parsing. https://github.com/bjascob/amrlib
- **amrlib-models:** Repository for models used by *amrlib* https://github.com/bjascob/amrlib-models (recommended: `parse_xfm_bart_large`)
- **Wikified parse_xfm_bart_large model:** https://cs.taltech.ee/staff/martin.verrev/model_parse_xfm_bart_large_wiki-v1_0_0.tgz

Thank you.